The changes to the drawings proposed above are made to replace informal drawings with formal drawings. In addition, figures 1 and 2 have been amended to correct obvious typographical errors. Namely, the call-out numbers 50, 60, and 70 have been replaced with the call-out numbers 250, 260, 270, respectively. The changes to the specification are made to correct obvious typographical and grammatical errors. No new matter is added.

Attached hereto is a marked-up version of the specification proposed by the current amendment. The attached page is captioned "Marked Up Version of the Prior Specification". In addition, attached hereto is a marked-up version of figures 1 and 2.

If for any reason the Examiner feels that consultation with Applicant's attorney would be helpful in the advancement of the prosecution, she is invited to call the telephone number below for an interview.

If there are any charges due with respect to this Amendment or otherwise, please charge them to Deposit Account No. 09-0463 maintained by Applicant' s assignee.

Respectfully submitted,

MARK R. GAMBINO

CANTOR COLBURN LLP
Applicant's Attorneys

By____________________________
David A. Fox, Esq.
Registration No. 38,807
Customer No. 23413

Date: February 8, 2002

Address 55 Griffin Road South, Bloomfield, CT 06002

Telephone: (860) 286-2929

MARKED UP VERSION OF THE PRIOR PENDING SPECIFICATION

[0011] Figures 4a and 4b [is] are a flow chart describing the steps for an SSL daemon to manage shared SSL sessions.

[0014] Figures 1 and 2 provide an overview of the elements that comprise an exemplary embodiment of the present invention. Figure 1 depicts an SSL daemon, which manages shared SSL sessions, receives requests for shared sessions, and acts as a shared session's link to applications (described below). The SSL daemon 250 is comprised of an SSL daemon process 9 ("daemon process"), a shared session 10, ("shared session") and a TCP/IP stack 11. The daemon process 9 includes program code for managing shared sessions 10 and may encompass queue management techniques such as "first task in the queue - first task out of the queue" ("FIFO") and highest-priority-first as well as any other software management techniques known to one of ordinary skill in the art. The daemon process 9 also includes program code for communicating with SSL wrapper processes (described below) and shared sessions 10 using SSL application programming interface ("API") 270 calls.

[0017] Figure 2 depicts the elements that comprise an application 260 enabled for SSL secured communications and enhanced by the present invention. Application 260 comprises at least one application process 5, at least one SSL wrapper process ("wrapper") 6, at least one SSL session 7 ("unshared session"), and a TCP/ IP stack 8. Where an application process 5 requests an unshared session, the application process 5 utilizes the unshared session 7 and the TCP/IP stack 8 that are tied to the application process 5. Where an application process 5 requests a shared session the application process 5 utilizes the shared session and TCP/IP stack that are tied to the daemon process (described in Figure 1).

[0018] The wrapper process 6 includes program code for receiving requests for SSL sessions, for determining whether the request is for a shared SSL session or an unshared SSL session, for passing requests for unshared SSL sessions to the unshared SSL session 7 tied to the application process 5, and for passing requests for shared SSL sessions to a daemon process. The wrapper process 6 also includes program code for communicating with application process 5, daemon processes, and unshared sessions 7 via SSL API 270 calls.

[0030] Figures 4a and 4b comprise [is] a flow chart showing steps for sharing an SSL session. In figure 4a, a[A]n application process passes an SSL_write API call to an SSL wrapper process 150. The SSL wrapper process receives the SSL_write API call and any accompanying input parameters 152 and uses the input parameter to determine whether the application process is requesting a shared or unshared SSL session 154.

[0031] If an application process has requested a shared SSL session, the SSL wrapper process passes the SSL_write API call to an SSL daemon process 156. The SSL daemon process passes the SSL_write API call to a shared SSL session 158. The shared SSL session encrypts the data 160 and passes the encrypted data to a TCP/IP stack 162. The TCP/IP stack packages and sends the encrypted data over a data network 164. As shown in figure 4b [In addition], the SSL session generates a return code and passes the return code to the SSL daemon process which in turn passes a second return code to the SSL wrapper process 166. The SSL wrapper process passes a second return code to the application process 168.

[0034] Returning to figure 4a: If an SSL wrapper process receives a request for a separate unshared SSL session, the wrapper process passes the SSL_write API call and any accompanying data/input parameters to an SSL session tied to the application process 182. The SSL session encrypts the data 184 and passes the encrypted data to the TCP/IP stack 186. The TCP/IP stack packages and sends the encrypted data over a data network 188. The SSL wrapper receives a return code from the SSL session and passes the return code to the application process 190.

[0035] In figure 4b, t[T]he application process passes an SSL_read API call to the wrapper, the wrapper passes the SSL_read API call to the unshared SSL session, the unshared SSL session reformats the SSL_read API call into a socket read API call and passes the socket read API call to the TCP/IP stack 191.